Supplementary Information

Antioxidant Activity of NSAIDs-Se Derivatives: Predictive QSAR-Machine Learning Models

Zhihui Fu[a], Amphawan Wiriyarattanakul[b], Wanting Xie[a], Pattamon Jantorn[c,d], Borwornlak Toopradab[c,d], Liyi Shi[a,e], Thanyada Rungrotmongkol[c,d]*, Phornphimon Maitarad[a]*

[a] Research Center of Nano Science and Technology, College of Sciences, Shanghai University, Shanghai 200444, PR China

[b] Program in Chemistry, Faculty of Science and Technology, Uttaradit Rajabhat University, Uttaradit 53000, Thailand

[c] Center of Excellence in Structural and Computational Biology, Department of Biochemistry, Chulalongkorn University, Bangkok 10330, Thailand

[d] Program in Bioinformatics and Computational Biology, Graduate School, Chulalongkorn University, Bangkok 10330, Thailand

[e] Emerging Industries Institute Shanghai University, Jiaxing, Zhejiang 314006, PR China

* Corresponding authors pmaitarad@shu.edu.cn (P.M.) t.rungrotmongkol@gmail.com (T.R.)

Table S1. 87 descriptors used for construction of QSAR models。

NO.	Descriptors	NO.	Descriptors	NO.	Descriptors
X1	Connolly surface area	X30	Chi (0)	X59	Molecular area (vdW area)
X2	Connolly surface occupied volume	X31	Chi (1)	X60	Molecular volume (vdW area)
X3	Solvent surface area	X32	Chi (2)	X61	Molecular density
X4	Solvent surface occupied volume	X33	Chi (3): path	X62	Principal moments of inertia (magnitude)
X5	Total molecular mass	X34	Chi (3): cluster	X63	Principal moment of inertia X
X6	Atom count	X35	Chi (0) (valence modified)	X64	Principal moment of inertia Y
X7	Element count	X36	Chi (1) (valence modified)	X65	Principal moment of inertia Z
X8	Rotatable bonds	X37	Chi (2) (valence modified)	X66	Radius of gyration
X9	Hydrogen bond donor	X38	Chi (3): path (valence modified)	X67	Ellipsoidal volume
X10	Hydrogen bond acceptor	X39	Chi (3): cluster (valence modified)	X68	Shadow area: XY plane
X11	AlogP	X40	Information content (IC)	X69	Shadow area: YZ plane
X12	AlogP98	X41	Bond information content (BIC)	X70	Shadow area: ZX plane
X13	Molecular refractivity	X42	Complementary information content (CIC)	X71	Shadow area fraction: XY plane
X14	Molecular flexibility	X43	Structural information content (SIC)	X72	Shadow area fraction: YZ plane
X15	Balaban index JX	X44	Edge adjacency/magnitude	X73	Shadow area fraction: ZX plane
X16	Balaban index JY	X45	Edge distance/magnitude	X74	Shadow length: LX
X17	Wiener index	X46	Vertex adjacency/equality	X75	Shadow length: LY
X18	Zagreb index	X47	Vertex adjacency/magnitude	X76	Shadow length: LZ
X19	Kappa-1	X48	Vertex distance/equality	X77	Shadow ratio
X20	Kappa-2	X49	Vertex distance/magnitude	X78	Total dipole
X21	Kappa-3	X50	Atomic composition (total)	X79	Dipole x
X22	Kappa-1 (alpha modified)	X51	E-state keys (sums): S_aaCH	X80*	**Dipole y**
X23	Kappa-2 (alpha modified)	X52	E-state keys (sums): S_aasC	X81	Dipole z
X24	Kappa-3 (alpha modified)	X53	E-state keys (sums): S_dO	X82*	**HOMO eigenvalue**
X25	Subgraph counts (0): path	X54*	**E-state keys (sums): S_dssC**	X83	LUMO eigenvalue
X26	Subgraph counts (1): path	X55*	**E-state keys (sums): S_sCH3**	X84	Mean polarizability
X27	Subgraph counts (2): path	X56	E-state keys (sums): S_ssCH2	X85	Total energy
X28	Subgraph counts (3): path	X57	Methoxy (Fragment Counts)	X86	Electronic energy
X29	Subgraph counts (3): cluster	X58	Methyl (Fragment Counts)	X87*	**Heat of formation**

Eighty-seven molecular descriptors were calculated for a set of thirty-six NSAID-Se derivatives. * represent selected descriptors to generate models.

Table S2. The definitions of 87 descriptors。

NO.	Definitions
X1-X2	Connolly surface: is at the boundary between the Connolly probe and the atoms (as represented by their scaled vdW radii), not at the locus of the probe center. Surface area: The surface area of an atom volume surface. Occupied volume: The volume on the atom side of the atom volume surface.
X3-X4	Solvent surface: The surface which is the locus of the probe center as the probe rolls over the scaled vdW surface.
X5-X6	Total mass of the whole molecular.
	Total number of atoms in the structure.
X7-X8	Total number of atoms of specified type in the structure.
	Number of rotatable bonds.
X9-X10	Number of hydrogen-bond donors.
	Number of hydrogen-bond acceptors.
X11-X12	Log of the partition coefficient (In this atom-based approach, each atom in the molecule is assigned to a particular class, with additive contributions to the total value of logP and the molar refractivity. For more information, see Leffler and Grunwald (1963))
X13	The molecular refractivity (m3/mol) index of a substituent is a combined measure of its size and polarizability.
X14	This descriptor is based on the structural properties that prevent a molecule from being "infinitely flexible", the model for which is an endless chain of C(sp3) atoms.
X15-X16	This is a highly discriminating descriptor whose values do not increase substantially with molecule size or the number of rings present (Balaban, 1982; Balaban and Ivanciuc, 1989).
X17	The Wiener index is the sum of the chemical bonds existing between all pairs of heavy atoms in the molecule. In graph-theoretical terms: the sum of lengths of minimal paths between all pairs of vertices representing heavy atoms. This is equal to half the sum of all D-matrix entries.
X18	The Zagreb index is defined as the sum of the squares of vertex valencies.
X19-X21	These indices compare the molecule graph with "minimal" and "maximal" graphs, where the meaning of "minimal" and "maximal" depends on the order, n. These are intended to capture different aspects of the molecular shape.
X22-X24	These indices are refinements of the shape indices described above that take into consideration the contribution made by covalent radii and hybridization to the shape of the molecule.
X25-X39	"This index, refined by Kier and Hall (1976), is a series of numbers designated by order and subgraph type. There are four subgraph types: Path, Cluster, Path/Cluster, and Chain. These types emphasize different aspects of atom connectivity within a molecule (the amount of branching ring structures present and flexibility). Here, these subgraph types are referred to as P, C, PC, and CH, respectively. They are defined as follows: Given a connected subgraph: If G contains a cycle, it is of type CH (chain). Otherwise: If all vertex valencies of G (valencies with respect to G, not the entire graph) are either greater than 2 or equal to 1, G is of type C (cluster). Otherwise: If all vertex valencies (as above) are either equal to 2 or 1, G is of type P (path). Otherwise: G is of type PC (Path/Cluster). That means the valencies greater than 2, equal to 2, and equal to 1, are all present. The order refers to the number of edges in a subgraph. Allowable orders are 0, 1, ..., M (M = the number of edges in the entire graph)."

X40-X43	"To determine information-content descriptors, molecules are viewed as structures that can be partitioned into subsets of elements that are in some sense equivalent. The notion of equivalence depends on the particular descriptor. Consider a partition of a set of N elements into k subsets each consisting of Nk elements Multigraph information content indices (IC, BIC, CIC, SIC) To each vertex v, an unordered sequence of ordered pairs, called a coordinate, is assigned: { (m1, n1), (m2, n2), ... , (mk, nk) } Where k is the valence of the vertex. There is one ordered pair (mj, nj) per each neighboring vertex, vj) and for every j = 1, ..., k: The valence of vj is nj The bond between v and vj is of order mj Having assigned the coordinates to vertices, the partition of vertices is constructed in the usual way, where two vertices are considered equivalent if their coordinates are the same (as unordered k-tuples, i.e., the repetitions of ordered pairs are not ignored, as they would be if the k-tuples were treated purely as sets). The index corresponding directly to this partition is the index IC (""Information Content""). The following indices are normalizations of IC: BIC = IC / lb(number of bonds counting bond orders) - Bonding Information Content SIC = IC / lb(number of vertices) - Structural Information Content The CIC (Complementary Information Content) measures the deviation of IC from its maximum possible value corresponding to the partition into classes containing one element each: $IC_{max} = -N \times (1/N) \times lb(1/N) = lb(N)$ and thus the CIC index is defined as: $CIC = lb(N) - IC$"
X44-X50	"Distance measurement - Each of the distance measurements found in the structure is listed in the form of an array of name/value pairs Distance count - The total number of distance measurements found in the structure Angle measurement - Each of the angle measurements found in the structure is listed in the form of an array of name/value pairs Angle count - The total number of angle measurements found in the structure Torsion measurement - Each of the torsion measurements found in the structure is listed in the form of an array of name/value pairs Torsion count - The total number of torsion measurements found in the structure"
X51-X56	"Electrotopological state keys are numerical values, computed for each atom in a molecule, which encode information about both the topological environment of that atom and the electronic interactions due to all other atoms in the molecule (Hall and Kier, 1995). The topological part of the relationship is based on the graph distances between each atom and all other atoms in the molecule. The electronic part is based on an intrinsic state value, plus perturbation due to intrinsic state differences between atoms in the molecule. In Materials Studio QSAR, E-states with the following symbols are calculated for each molecule: I_xxx - Indicator for type xxx (presence/absence, 0/1) N_xxx - Counter for type xxx (number of times it appears in the molecule) S_xxx - Sum for type xxx (the actual E-state key as defined by the formulas above) where the xxx types are: s - single bond d - double bond t - triple bond a - aromatic bond For example, S_aasC is the sum descriptor for carbon with two aromatic bonds and one single bond. N_sNH2 is the counter descriptor for a N bonded to two hydrogens and a single bond."
X57-X58	The fragment counting model allows you to determine the number of a specified fragments contained within a structure, using the pattern matching algorithm in Materials Visualizer. The algorithm searches for matches based on topology, element type and bond order. Hydrocarbons - Fragment counts for a range of hydrocarbon fragments Functional groups - Fragment counts for a range of functional groups
X59	Molecular area (vdW area) - describes the van der Waals area of a molecule. The molecular surface area determines the extent to which a molecule exposes itself to the external environment. This descriptor is related to binding, transport, and solubility.

X60	Molecular volume (vdW volume) - describes the volume inside the van der Waals area of a molecule. Molecular volume is related to binding and transport.
X61	Molecular density - defined as the ratio of molecular weight to molecular volume. It has units of g ml-1. The molecular density reflects the types of atoms and how tightly they are packed in a molecule. It can be related to transport and melt behavior.
X62-X65	"These descriptors calculate the principal moments of inertia of a molecule according to the following rules: The moment of inertia of a molecule about a given axis is given by: $$I = \sum_i m_i d_i^2$$ where di is the perpendicular distance from each atom to the axis. The principal moments of inertia of a molecule are the moments of inertia about each of the principal axes of the molecule. The principal moments of inertia and the principal axes of the molecule are the eigenvalues and eigenvectors of the inertia tensor of the system. If all three moments are equal, the molecule is considered to be a spherical top. If two moments are equal, the molecule is considered to be a symmetrical top. If no moments are equal, the molecule is considered to be an asymmetrical top. Principal moments of inertia (magnitude) - the magnitude of the principal moments of inertia Principal moment of inertia X - the moment of inertia about the longest principal axis Principal moment of inertia Y - the moment of inertia about the intermediate principal axis Principal moment of inertia Z - the moment of inertia about the shortest principal axis These descriptors are calculated in units of amu Å2 For more information about this descriptor, see Hill (1960)."
X66	where N is the number of atoms and x, y, and z are the atomic coordinates relative to the center of mass. $$Rog = \sqrt{(\Sigma \quad {}_N \quad)}$$
X67	Ellipsoidal volume - describes the volume of the ellipsoid of inertia derived from the inertia tensor of the system. This has axes proportional to the inverse of the square root of the principal moments of inertia, aligned along the principal axes.
X68-X77	This set of geometric descriptors helps to characterize the shape of the molecules. The descriptors are calculated by projecting the molecular surface on three mutually perpendicular planes, xy, yz, and xz (Rohrbaugh and Jurs, 1987). These descriptors depend not only on conformation, but also on the orientation of the molecule. To calculate them, the molecules are first rotated to align the principal moments of inertia with the x, y, and z axes. A total of 10 descriptors are calculated in this set: Shadow area: XY plane -area of the molecular shadow in the xy plane (Sxy). Shadow area: YZ plane -area of the molecular shadow in the yz plane (Syz). Shadow area: ZX plane -area of the molecular shadow in the xz plane (Sxz). Shadow area fraction: XY plane - fraction of area of molecular shadow in the xy plane over area of enclosing rectangle (Sxy,f). Shadow area fraction: YZ plane - fraction of area of molecular shadow in the yz plane over area of enclosing rectangle (Syz,f). Shadow area fraction: ZX plane - fraction of area of molecular shadow in the xz plane over area of enclosing rectangle (Sxz,f). Shadow length: LX -length of molecule in the x dimension (Lx). Shadow length: LY -length of molecule in the y dimension (Ly). Shadow length: LZ -length of molecule in the z dimension (Lz). Shadow ratio -ratio of largest to smallest dimension (δ)
X78-X81	These descriptors calculate the molecular dipole moments from partial charges defined on the atoms of the molecule. If no partial charges are defined, the molecular dipole moment will be zero. Dipole moment (magnitude) - the magnitude of the dipole moment Dipole moment X - the dipole moment about the X axis Dipole moment Y - the dipole moment about the Y axis Dipole moment Z - the dipole moment about the Z axis
X82-X87	The VAMP electrostatics model allows to calculate a variety of descriptors using the VAMP semiempirical molecular orbital code. Energy - Total energy (eV), electronic energy (eV), and heat of formation (kcal/mol) Orbitals - HOMO eigenvalue and LUMO eigenvalue (eV)

Table S3. The datasets of 36 studied compounds consisted of their Log[%DPPH] and molecular descriptors.

Structures	Log[%DPPH]	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
cpd1	1.326	423.972	461.157	692.706	1256.065	501.478	54	24	10	1	4
cpd2	1.423	328.119	362.770	557.350	990.437	386.357	45	19	9	2	2
cpd3	1.695	385.247	424.755	640.654	1155.426	502.860	53	23	11	1	5
cpd4	1.483	348.140	382.812	573.262	1038.510	399.352	45	20	10	1	3
cpd5	1.354	360.770	348.497	633.178	1052.946	375.330	43	18	9	1	3
cpd6	1.389	382.249	413.027	639.973	1138.581	432.426	54	21	10	2	3
cpd7	1.784	339.331	344.480	584.257	1002.412	351.352	45	17	10	1	2
cpd8	1.717	266.533	284.394	471.036	805.652	325.226	33	13	9	1	4
cpd9	1.516	431.330	481.526	707.188	1291.272	544.465	56	24	9	1	6
cpd10	1.607	351.783	377.623	588.553	1045.750	432.319	45	19	8	1	5
cpd11	1.59	316.259	360.733	537.194	963.898	429.344	47	19	8	2	4
cpd12	1.412	411.137	466.236	663.501	1230.216	545.847	55	23	10	1	7
cpd13	1.685	358.160	391.341	587.413	1061.050	442.339	47	20	9	1	5
cpd14	1.571	360.135	366.818	621.699	1061.864	418.317	45	18	8	1	5
cpd15	1.428	401.823	430.071	667.655	1190.362	475.413	56	21	9	2	5
cpd16	1.838	324.694	364.095	545.797	979.583	394.339	47	17	9	1	4
cpd17	1.86	287.010	296.026	505.461	855.802	368.213	35	13	8	1	6
cpd18	1.494	332.133	361.361	559.725	994.681	366.363	47	18	11	0	3
cpd19	1.646	450.976	463.926	748.474	1316.138	517.871	55	24	12	0	6
cpd20	1.471	360.851	376.782	600.000	1058.125	401.368	47	20	10	1	3
cpd21	1.483	388.070	426.976	642.384	1158.120	447.437	56	22	11	1	4
cpd22	1.391	347.419	355.186	604.510	1029.293	433.328	47	19	9	0	6
cpd23	1.712	368.763	381.998	629.165	1093.143	441.311	44	19	10	1	6
cpd24	1.66	460.023	473.854	765.545	1342.057	516.489	56	25	11	0	5
cpd25	1.757	406.120	380.725	716.586	1174.646	414.363	47	21	11	0	4
cpd26	1.436	274.787	298.290	482.778	833.544	340.237	35	14	10	0	5
cpd27	1.866	369.102	383.842	633.554	1093.757	447.330	47	20	9	0	6
cpd28	1.686	333.824	371.499	559.822	1005.289	409.350	49	18	10	0	5
cpd29	1.563	459.660	482.946	757.061	1345.291	560.858	57	24	11	0	8
cpd30	1.367	384.393	393.538	650.604	1128.415	444.355	49	20	9	1	5
cpd31	1.617	399.901	437.678	662.314	1191.304	490.424	58	22	10	1	6
cpd32	1.568	341.714	369.234	582.533	1025.479	390.341	45	19	10	0	4
cpd33	1.836	353.615	398.450	590.473	1069.566	484.298	46	19	9	1	8
cpd34	1.652	465.056	488.434	768.722	1365.834	559.476	58	25	10	0	7
cpd35	1.822	406.591	395.124	704.546	1182.173	457.350	49	21	10	0	6
cpd36	1.511	273.660	272.728	492.027	815.012	369.197	34	13	8	0	7

Structures	Log[%DPPH]	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20
cpd1	1.326	4.328	4.688	124.902	7.562	1.808	1.850	3001	156	25.620	12.459
cpd2	1.423	3.472	4.339	104.032	6.387	2.037	2.079	1507	114	20.314	10.871
cpd3	1.695	2.084	3.614	123.955	7.454	1.834	1.900	2854	156	25.620	12.459
cpd4	1.483	3.002	3.718	105.163	6.912	1.973	2.009	1752	118	21.302	11.658
cpd5	1.354	2.530	4.853	97.501	6.131	2.028	2.075	1414	110	19.326	10.096
cpd6	1.389	1.432	3.794	112.285	6.178	1.751	1.803	1931	138	21.703	10.156
cpd7	1.784	3.371	3.964	93.379	7.531	2.399	2.434	1162	94	19.048	10.680
cpd8	1.717	0.785	1.584	76.596	6.267	2.462	2.568	824	84	17.053	9.834
cpd9	1.516	5.966	6.792	125.512	7.917	1.718	1.775	3645	170	27.585	12.301
cpd10	1.607	5.243	6.141	100.549	6.774	1.852	1.918	2027	128	22.291	10.519
cpd11	1.59	5.111	6.443	104.641	6.661	1.93	1.997	1923	128	22.291	10.519
cpd12	1.412	3.723	5.718	124.564	7.812	1.742	1.821	3480	170	27.585	12.301
cpd13	1.685	4.641	5.822	105.772	7.16	1.875	1.934	2214	132	23.281	11.253
cpd14	1.571	4.169	6.957	98.110	6.419	1.901	1.972	1818	124	21.302	9.796
cpd15	1.428	4.099	5.987	113.229	6.562	1.66	1.727	2429	152	23.659	10.080
cpd16	1.838	5.010	6.068	93.988	7.592	2.305	2.380	1514	108	21.043	10.096
cpd17	1.86	2.423	3.688	77.205	6.356	2.342	2.483	1106	98	19.048	9.209
cpd18	1.494	4.185	4.857	96.004	8.468	2.305	2.353	1361	98	20.045	11.523
cpd19	1.646	3.130	4.718	126.606	8.179	1.761	1.829	3210	160	26.602	13.185
cpd20	1.471	4.054	5.021	106.632	7.184	1.944	1.994	1744	118	21.302	11.658
cpd21	1.483	2.334	4.763	115.090	6.876	1.667	1.722	2215	142	22.680	10.858
cpd22	1.391	4.983	7.850	100.735	7.172	1.812	1.889	2091	128	22.291	10.519
cpd23	1.712	4.003	4.991	102.523	7.381	1.939	2.026	2168	130	23.281	11.870
cpd24	1.66	5.374	5.792	127.554	8.291	1.736	1.782	3366	160	26.602	13.185
cpd25	1.757	3.817	4.611	107.788	7.726	1.889	1.931	2014	122	22.291	12.457
cpd26	1.436	1.367	2.266	79.195	7.185	2.346	2.461	984	88	18.050	10.688
cpd27	1.866	6.058	7.205	103.174	7.536	1.798	1.871	2320	132	23.281	11.253
cpd28	1.686	5.824	6.961	96.613	8.463	2.223	2.309	1753	112	22.042	10.871
cpd29	1.563	4.769	6.821	127.215	8.515	1.673	1.754	3896	174	28.569	12.992
cpd30	1.367	5.693	7.125	107.241	7.419	1.847	1.920	2206	132	23.281	11.253
cpd31	1.617	3.972	6.867	115.699	7.239	1.57	1.640	2765	156	24.639	10.745
cpd32	1.568	3.345	5.746	100.126	6.923	1.922	1.976	1643	114	20.314	10.871

cpd33	1.836	5.641	7.745	103.132	7.664	1.861	1.966	2698	144	25.262	11.571
cpd34	1.652	7.013	7.896	128.163	8.624	1.651	1.711	4070	174	28.569	12.992
cpd35	1.822	5.455	6.714	108.397	7.933	1.8	1.865	2524	136	24.271	12.000
cpd36	1.511	3.073	4.335	75.255	6.513	2.326	2.489	1106	98	19.048	9.209
Structures	Log[%DPPH]	X21	X22	X23	X24	X25	X26	X27	X28	X29	X30
cpd1	1.326	6.992	22.670	10.340	5.574	31	33	45	58	10	22.380
cpd2	1.423	6.353	17.552	8.733	4.848	24	25	32	40	6	17.364
cpd3	1.695	6.533	22.543	10.251	5.134	31	33	45	60	10	22.380
cpd4	1.483	6.910	18.409	9.386	5.287	25	26	33	41	6	18.071
cpd5	1.354	5.786	16.981	8.305	4.542	23	24	31	39	6	16.656
cpd6	1.389	4.776	19.442	8.579	3.863	27	29	40	56	10	19.278
cpd7	1.784	7.705	17.232	9.178	6.442	21	21	26	29	5	15.665
cpd8	1.717	6.817	14.869	8.008	5.319	19	19	23	26	4	14.088
cpd9	1.516	8.000	24.923	10.482	6.646	33	35	50	60	14	24.173
cpd10	1.607	7.510	19.948	8.829	6.146	26	27	37	42	10	19.156
cpd11	1.59	7.510	19.820	8.738	6.073	26	27	37	42	10	19.156
cpd12	1.412	7.492	24.796	10.397	6.146	33	35	50	62	14	24.173
cpd13	1.685	8.100	20.679	9.348	6.549	27	28	38	43	10	19.864
cpd14	1.571	6.910	19.247	8.338	5.742	25	26	36	41	10	18.449
cpd15	1.428	5.627	21.683	8.776	4.767	29	31	45	58	14	21.071
cpd16	1.838	9.157	19.527	8.942	8.048	23	23	31	31	9	17.458
cpd17	1.86	8.265	17.162	7.777	6.894	21	21	28	28	8	15.880
cpd18	1.494	9.037	18.389	10.131	7.803	22	22	27	29	5	16.372
cpd19	1.646	7.014	23.678	11.053	5.643	32	34	46	61	10	23.087
cpd20	1.471	7.260	18.695	9.606	5.736	25	26	33	40	6	18.071
cpd21	1.483	5.202	20.571	9.358	4.316	28	30	41	57	10	19.985
cpd22	1.391	7.881	20.393	9.144	6.729	26	27	37	41	10	19.156
cpd23	1.712	8.490	20.462	9.739	6.756	27	28	37	42	9	19.700
cpd24	1.66	7.498	23.806	11.144	6.114	32	34	46	59	10	23.087
cpd25	1.757	7.881	19.553	10.273	6.239	26	27	34	41	6	18.778
cpd26	1.436	8.148	16.026	8.967	6.645	20	20	24	26	4	14.795
cpd27	1.866	8.490	21.094	9.646	7.134	27	28	38	42	10	19.864
cpd28	1.686	10.576	20.684	9.820	9.524	24	24	32	31	9	18.165
cpd29	1.563	7.998	25.934	11.164	6.691	34	36	51	63	14	24.880
cpd30	1.367	8.490	20.966	9.554	7.057	27	28	38	42	10	19.864
cpd31	1.617	6.081	22.817	9.518	5.261	30	32	46	59	14	21.778
cpd32	1.568	6.682	18.123	9.167	5.432	24	25	32	39	6	17.364
cpd33	1.836	9.777	22.736	9.776	8.146	29	30	42	44	13	21.493
cpd34	1.652	8.531	26.062	11.250	7.222	34	36	51	61	14	24.880
cpd35	1.822	9.140	21.826	10.178	7.594	28	29	39	43	10	20.571
cpd36	1.511	8.265	17.322	7.896	7.008	21	21	28	28	8	15.880
Structures	Log[%DPPH]	X31	X32	X33	X34	X35	X36	X37	X38	X39	X40
cpd1	1.326	14.901	13.237	10.583	2.018	19.935	13.462	10.420	7.365	1.115	4.285
cpd2	1.423	11.630	9.652	7.943	1.133	15.640	9.913	7.313	5.124	0.537	3.605
cpd3	1.695	14.956	12.938	10.854	1.852	19.744	12.163	9.196	6.540	0.851	4.107
cpd4	1.483	12.130	9.983	8.296	1.121	15.780	10.183	7.472	5.170	0.517	3.574
cpd5	1.354	11.151	9.220	7.744	1.132	15.048	9.585	7.124	5.056	0.534	3.708
cpd6	1.389	13.147	11.053	9.643	1.591	17.774	11.761	8.783	6.727	0.831	4.06
cpd7	1.784	10.041	8.450	6.100	1.207	14.769	9.485	7.491	4.609	0.826	3.463
cpd8	1.717	9.147	7.518	5.236	0.947	12.224	7.782	5.484	3.463	0.265	3.577
cpd9	1.516	15.548	15.194	10.906	3.579	20.622	13.805	11.362	7.880	1.597	4.173
cpd10	1.607	12.277	11.609	8.259	2.694	15.859	10.178	8.199	5.578	0.991	3.642
cpd11	1.59	12.277	11.609	8.266	2.694	16.326	10.257	8.255	5.639	1.018	3.427
cpd12	1.412	15.602	14.895	11.177	3.413	20.431	12.506	10.139	7.055	1.333	3.957
cpd13	1.685	12.777	11.940	8.620	2.681	16.467	10.526	8.415	5.685	0.999	3.588
cpd14	1.571	11.798	11.177	8.067	2.693	15.734	9.928	8.066	5.572	1.016	3.703
cpd15	1.428	13.794	13.010	9.967	3.152	18.461	12.104	9.726	7.242	1.313	4.116
cpd16	1.838	10.687	10.407	6.424	2.768	15.456	9.828	8.433	5.124	1.307	3.469
cpd17	1.86	9.794	9.475	5.559	2.507	12.911	8.125	6.427	3.978	0.747	3.404
cpd18	1.494	10.524	8.931	6.076	1.303	15.385	9.847	7.804	4.689	0.885	3.482
cpd19	1.646	15.456	13.280	11.172	1.852	20.360	12.552	9.390	6.690	0.838	4.093
cpd20	1.471	12.113	10.111	8.053	1.217	16.255	10.260	7.565	5.249	0.575	3.513
cpd21	1.483	13.647	11.388	10.023	1.591	18.389	12.150	8.977	6.922	0.818	3.964
cpd22	1.391	12.281	11.659	8.042	2.789	16.350	10.291	8.379	5.651	1.076	3.796
cpd23	1.712	12.897	11.680	8.216	2.336	15.966	10.154	7.397	4.844	0.534	3.708
cpd24	1.66	15.401	13.579	10.901	2.018	20.550	13.851	10.613	7.515	1.102	4.164
cpd25	1.757	12.613	10.465	8.272	1.217	16.395	10.545	7.785	5.250	0.577	3.671
cpd26	1.436	9.630	7.978	5.346	1.030	12.839	8.128	5.737	3.584	0.303	3.484
cpd27	1.866	12.760	12.091	8.234	2.790	16.474	10.540	8.512	5.658	1.051	3.856
cpd28	1.686	11.171	10.889	6.399	2.864	16.071	10.190	8.746	5.204	1.367	3.491
cpd29	1.563	16.102	15.237	11.496	3.413	21.046	12.895	10.333	7.205	1.319	3.951
cpd30	1.367	12.760	12.069	8.376	2.777	16.942	10.603	8.508	5.764	1.056	3.352
cpd31	1.617	14.294	13.345	10.347	3.152	19.076	12.493	9.920	7.437	1.300	3.961

cpd32	1.568	11.635	9.702	7.719	1.229	15.663	9.947	7.437	5.136	0.594	3.804
cpd33	1.836	13.544	13.637	8.539	3.897	16.653	10.497	8.339	5.359	1.016	3.814
cpd34	1.652	16.048	15.536	11.224	3.579	21.237	14.194	11.556	8.030	1.584	4.065
cpd35	1.822	13.260	12.422	8.595	2.777	17.082	10.889	8.727	5.765	1.059	3.682
cpd36	1.511	9.794	9.475	5.559	2.507	12.819	8.014	6.307	3.869	0.738	3.404
Structures	Log[%DPPH]	X41	X42	X43	X44	X45	X46	X47	X48	X49	X50
cpd1	1.326	0.782	0.670	0.865	584.267	10649.140	346.895	398.930	3756.648	9227.623	92.683
cpd2	1.423	0.709	0.980	0.786	384.000	5607.993	245.213	282.193	2105.241	5102.615	69.429
cpd3	1.695	0.75	0.847	0.829	584.267	10662.090	346.895	398.930	3664.226	9241.886	91.921
cpd4	1.483	0.691	1.070	0.77	398.930	6128.751	258.347	296.423	2365.620	5600.784	70.256
cpd5	1.354	0.726	0.815	0.82	369.160	5079.442	232.192	268.078	1981.367	4603.396	67.832
cpd6	1.389	0.779	0.695	0.854	505.754	7904.048	292.060	339.763	2679.179	6708.208	83.388
cpd7	1.784	0.728	0.929	0.788	296.423	3714.704	200.089	226.477	1643.631	3719.108	65.607
cpd8	1.717	0.761	0.671	0.842	254.084	2946.454	175.251	199.421	1256.661	2949.105	58.301
cpd9	1.516	0.762	0.871	0.827	664.386	12177.950	374.836	429.050	4357.715	10648.070	101.131
cpd10	1.607	0.716	1.058	0.775	459.500	6672.883	271.591	310.764	2622.488	6123.250	77.712
cpd11	1.59	0.674	1.274	0.729	459.500	6696.578	271.591	310.764	2539.603	6141.171	81.362
cpd12	1.412	0.723	1.087	0.784	664.386	12189.470	374.836	429.050	4264.178	10661.720	104.316
cpd13	1.685	0.694	1.167	0.755	474.842	7270.108	284.941	325.212	2832.483	6691.799	81.434
cpd14	1.571	0.725	0.940	0.797	444.235	6113.738	258.347	296.423	2405.279	5588.351	78.881
cpd15	1.428	0.787	0.742	0.847	584.267	9205.253	319.295	369.160	3183.394	7905.635	95.836
cpd16	1.838	0.729	1.055	0.767	369.160	4593.310	225.475	254.084	2018.654	4600.345	76.785
cpd17	1.86	0.724	0.988	0.775	325.212	3723.531	200.089	226.477	1578.950	3728.937	68.606
cpd18	1.494	0.724	0.978	0.781	310.764	4140.313	212.717	240.215	1849.930	4145.089	67.911
cpd19	1.646	0.743	0.907	0.819	600.168	11406.970	360.824	413.947	3996.073	9933.385	93.894
cpd20	1.471	0.685	1.131	0.757	398.930	6136.723	258.347	296.423	2348.511	5605.793	72.520
cpd21	1.483	0.755	0.843	0.825	521.319	8536.118	305.631	354.413	2961.854	7288.980	85.744
cpd22	1.391	0.737	0.904	0.808	459.500	6668.595	271.591	310.764	2662.265	6117.673	78.607
cpd23	1.712	0.712	1.047	0.78	459.500	7285.636	284.941	325.212	2804.777	6701.124	81.263
cpd24	1.66	0.756	0.836	0.833	600.168	11395.770	360.824	413.947	4087.330	9920.247	94.321
cpd25	1.757	0.705	1.029	0.781	413.947	6683.754	271.591	310.764	2621.623	6130.111	71.833
cpd26	1.436	0.733	0.838	0.806	268.078	3321.415	187.598	212.877	1437.734	3324.459	59.619
cpd27	1.866	0.749	0.899	0.811	474.842	7254.644	284.941	325.212	2893.024	6679.048	78.189
cpd28	1.686	0.726	1.094	0.761	384.000	5070.381	238.359	268.078	2250.269	5077.750	77.211
cpd29	1.563	0.717	1.136	0.777	680.587	12989.460	388.930	444.235	4625.621	11407.950	105.639
cpd30	1.367	0.654	1.403	0.705	474.842	7278.164	284.941	325.212	2808.807	6696.876	83.820
cpd31	1.617	0.755	0.945	0.807	600.168	9892.040	333.051	384.000	3493.113	8540.287	97.540
cpd32	1.568	0.739	0.781	0.83	384.000	5581.379	245.213	282.193	2212.411	5080.331	69.429
cpd33	1.836	0.729	1.044	0.785	536.955	8535.760	311.942	354.413	3310.826	7900.073	86.377
cpd34	1.652	0.738	1.022	0.799	680.587	12979.790	388.930	444.235	4718.696	11395.550	100.873
cpd35	1.822	0.707	1.125	0.766	490.261	7878.684	298.392	339.763	3115.200	7274.270	81.134
cpd36	1.511	0.724	0.988	0.775	325.212	3723.531	200.089	226.477	1578.950	3728.937	64.007
Structures	Log[%DPPH]	X51	X52	X53	X54	X55	X56	X57	X58	X59	X60
cpd1	1.326	11.432	3.701	24.160	2.911	3.656	2.448	0	2	506.222	429.795
cpd2	1.423	13.603	4.814	12.429	-0.090	4.138	2.304	0	2	385.432	340.082
cpd3	1.695	12.744	2.827	26.024	1.117	3.659	2.846	1	2	470.705	395.612
cpd4	1.483	16.349	2.039	24.810	-0.109	1.834	2.243	0	1	417.420	350.667
cpd5	1.354	0.000	0.000	12.346	0.241	3.991	2.575	1	2	408.288	330.038
cpd6	1.389	2.228	2.778	12.638	1.579	4.366	6.891	0	2	457.221	385.965
cpd7	1.784	8.352	2.376	12.068	0.066	6.348	3.529	0	3	397.376	321.805
cpd8	1.717	6.590	0.595	22.855	-0.738	1.290	2.098	0	1	318.122	265.551
cpd9	1.516	11.061	3.388	24.122	2.376	3.534	0.595	0	2	532.715	444.290
cpd10	1.607	13.750	1.317	12.119	-0.323	1.645	0.496	0	1	422.925	350.707
cpd11	1.59	12.992	4.284	12.399	-0.294	4.018	0.617	0	2	399.661	334.434
cpd12	1.412	12.309	2.545	25.981	0.727	3.517	0.993	1	2	503.434	435.820
cpd13	1.685	15.728	1.748	24.767	-0.391	1.709	0.556	0	1	434.901	364.526
cpd14	1.571	0.000	0.000	12.317	0.037	3.823	0.888	1	2	417.918	343.730
cpd15	1.428	2.238	2.922	12.674	-0.016	4.381	5.525	0	2	479.898	399.105
cpd16	1.838	7.972	2.167	12.038	-0.138	6.126	1.772	0	3	413.713	334.453
cpd17	1.86	6.173	0.302	22.810	-1.033	1.216	0.411	0	1	335.168	280.118
cpd18	1.494	8.540	2.519	11.790	-0.150	6.478	3.567	0	3	403.174	332.227
cpd19	1.646	12.846	2.901	25.655	0.946	3.703	3.152	1	2	524.756	439.463
cpd20	1.471	13.919	5.287	12.061	-0.236	4.161	2.190	0	2	415.979	355.273
cpd21	1.483	2.237	2.813	12.247	1.421	4.407	7.329	0	2	464.436	397.338
cpd22	1.391	0.000	0.000	11.961	-0.258	3.970	0.878	1	2	402.889	333.869
cpd23	1.712	11.762	0.704	11.977	-0.419	0.000	1.630	0	0	433.634	357.111
cpd24	1.66	11.520	3.815	23.831	2.939	3.693	2.653	0	2	537.541	446.081
cpd25	1.757	16.561	2.208	24.509	-0.275	1.951	2.165	0	1	442.136	365.749
cpd26	1.436	6.861	0.999	22.621	-0.793	1.312	1.858	0	1	334.442	279.721
cpd27	1.866	13.284	2.717	11.861	-0.445	1.849	0.343	0	1	428.334	363.338
cpd28	1.686	8.218	2.343	11.760	-0.353	6.288	1.654	0	3	406.382	345.295
cpd29	1.563	12.466	2.656	25.614	0.584	3.580	1.165	1	2	539.309	452.958
cpd30	1.367	13.398	4.843	12.031	-0.440	4.056	0.337	0	2	443.766	370.436

cpd31	1.617	2.188	2.687	12.217	1.126	4.289	5.046	0	2	472.823	409.967
cpd32	1.568	0.000	0.000	11.991	-0.054	4.112	2.731	1	2	410.566	342.974
cpd33	1.836	8.445	0.327	11.947	-0.622	0.000	-0.223	0	0	432.463	367.381
cpd34	1.652	11.196	3.546	23.794	2.457	3.587	0.666	0	2	553.192	459.561
cpd35	1.822	16.025	1.963	24.467	-0.546	1.847	0.312	0	1	449.832	377.397
cpd36	1.511	6.002	0.128	22.679	-1.311	1.196	-0.012	0	1	314.988	258.615
Structures	Log[%DPPH]	X61	X62	X63	X64	X65	X66	X67	X68	X69	X70
cpd1	1.326	1.167	11864.330	2623.574	8019.369	8340.794	4.702	971.632	119.609	61.514	87.963
cpd2	1.423	1.136	6584.136	1414.636	4288.189	4791.773	3.868	608.958	91.313	53.251	72.717
cpd3	1.695	1.271	10985.720	2653.909	6976.834	8060.188	4.155	946.699	100.150	64.305	83.126
cpd4	1.483	1.139	5548.074	2522.332	3316.408	3663.387	3.802	782.261	80.355	68.654	82.472
cpd5	1.354	1.137	12410.040	922.197	8584.177	8914.627	4.885	823.419	89.227	46.672	95.852
cpd6	1.389	1.12	9959.808	1467.620	6552.719	7355.660	4.678	1003.269	113.445	49.596	78.621
cpd7	1.784	1.092	6313.597	1371.603	4029.930	4662.604	4.297	773.137	90.186	52.938	74.147
cpd8	1.717	1.225	3419.963	1202.575	2090.695	2424.655	3.466	496.979	73.597	46.692	56.905
cpd9	1.516	1.225	12510.030	2881.180	8582.902	8633.269	4.698	1035.538	109.180	66.410	102.347
cpd10	1.607	1.233	7380.892	2477.761	4038.215	5659.602	4.228	830.199	98.386	55.094	78.896
cpd11	1.59	1.284	5525.331	1908.674	3131.014	4133.158	3.765	609.550	87.672	56.331	63.128
cpd12	1.412	1.252	11191.030	3756.080	6529.751	8276.075	4.174	963.616	113.079	70.894	81.602
cpd13	1.685	1.213	8029.836	2252.566	4770.892	6053.330	3.977	708.398	105.107	57.364	69.322
cpd14	1.571	1.217	11255.560	1369.571	7645.964	8145.623	4.826	908.209	101.609	48.917	79.950
cpd15	1.428	1.191	11745.130	1574.586	8129.116	8329.833	4.671	1019.123	103.507	57.030	98.118
cpd16	1.838	1.179	5720.396	1828.363	3285.740	4310.909	3.805	705.668	88.551	54.562	64.663
cpd17	1.86	1.314	5853.116	1031.285	3920.729	4221.765	3.638	613.022	78.912	45.060	68.009
cpd18	1.494	1.103	5541.202	1555.261	3363.830	4119.555	4.038	680.653	88.596	52.912	73.947
cpd19	1.646	1.178	17476.590	3215.494	11081.570	13125.960	5.199	2807.720	135.354	59.743	90.776
cpd20	1.471	1.13	6737.456	2482.118	3345.010	5295.593	4.21	916.266	110.626	59.810	62.192
cpd21	1.483	1.126	10153.270	1663.524	6770.813	7380.899	4.452	1248.566	110.263	56.878	81.837
cpd22	1.391	1.298	11429.030	1434.879	7652.056	8367.192	4.884	791.296	99.503	44.942	83.544
cpd23	1.712	1.236	8329.155	2299.787	5255.871	6038.346	4.148	927.130	95.966	68.150	82.650
cpd24	1.66	1.158	19881.480	2908.273	12925.210	14824.110	5.63	1708.515	137.341	55.883	95.026
cpd25	1.757	1.133	19515.200	1668.103	13377.720	14110.180	5.628	1346.570	97.923	57.637	106.846
cpd26	1.436	1.216	3916.443	1333.591	2337.156	2845.657	3.325	523.189	74.044	52.166	57.555
cpd27	1.866	1.231	13256.520	1215.370	9302.767	9365.721	4.867	899.365	96.625	49.563	98.523
cpd28	1.686	1.186	5883.331	1913.975	3419.154	4388.584	4.006	761.386	92.035	55.039	71.202
cpd29	1.563	1.238	19215.940	3716.878	12023.050	14521.820	5.245	2826.958	142.933	61.584	87.259
cpd30	1.367	1.2	10734.920	2099.773	6717.094	8106.173	4.855	1108.855	116.402	50.715	74.562
cpd31	1.617	1.196	11048.750	2439.888	7133.113	8077.162	4.457	1088.324	109.086	64.319	88.223
cpd32	1.568	1.138	5943.022	1688.646	3684.042	4346.932	4.271	841.593	85.669	59.884	80.723
cpd33	1.836	1.318	7879.308	2404.312	4717.520	5835.048	3.981	881.702	92.269	59.469	85.489
cpd34	1.652	1.217	21904.810	3382.877	14105.320	16413.920	5.658	1821.253	142.441	58.164	94.202
cpd35	1.822	1.212	15328.240	2758.649	9648.779	11586.450	0.638	1666.132	115.965	59.643	83.080
cpd36	1.511	1.428	4524.673	1172.599	2817.097	3340.904	0.615	636.116	75.467	46.582	53.980
Structures	Log[%DPPH]	X71	X72	X73	X74	X75	X76	X77	X78	X79	X80
cpd1	1.326	0.648	0.605	0.574	16.677	11.061	9.189	1.815	3.127	1.815	-1.321
cpd2	1.423	0.673	0.666	0.638	13.903	9.765	8.194	1.697	0.778	-0.674	-0.273
cpd3	1.695	0.560	0.665	0.656	15.318	11.684	8.270	1.852	8.009	-5.88	2.896
cpd4	1.483	0.537	0.617	0.672	12.843	11.654	9.550	1.345	4.866	-2.321	1.783
cpd5	1.354	0.502	0.636	0.593	19.768	8.985	8.170	2.420	8.856	7.148	-1.298
cpd6	1.389	0.734	0.630	0.570	16.462	9.393	8.379	1.965	2.591	0.862	-1.672
cpd7	1.784	0.592	0.584	0.592	14.510	10.492	8.639	1.680	4.344	2.918	-3.153
cpd8	1.717	0.676	0.655	0.620	11.840	9.202	7.751	1.528	7.294	-5.872	-0.84
cpd9	1.516	0.612	0.644	0.669	16.276	10.965	9.397	1.732	4.214	1.209	1.243
cpd10	1.607	0.576	0.692	0.730	15.224	11.225	7.098	2.145	3.826	2.259	-0.947
cpd11	1.59	0.695	0.680	0.610	12.548	10.048	8.247	1.522	4.284	2.254	2.941
cpd12	1.412	0.671	0.588	0.572	14.120	11.941	10.104	1.397	9.082	7.404	-5.176
cpd13	1.685	0.717	0.686	0.661	13.558	10.814	7.735	1.753	10.086	-0.102	-4.847
cpd14	1.571	0.686	0.641	0.504	17.554	8.444	9.037	2.079	2.131	0.369	2.098
cpd15	1.428	0.564	0.593	0.637	17.139	10.705	8.984	1.908	2.564	0.288	1.501
cpd16	1.838	0.665	0.723	0.684	12.913	10.315	7.316	1.765	5.914	2.299	-0.993
cpd17	1.86	0.609	0.618	0.576	14.493	8.943	8.147	1.779	9.139	-8.591	-1.092
cpd18	1.494	0.609	0.659	0.646	14.409	10.102	7.950	1.812	8.121	5.613	3.59
cpd19	1.646	0.545	0.538	0.549	19.239	12.910	8.594	2.239	5.508	-3.343	2.4
cpd20	1.471	0.660	0.667	0.650	13.382	12.533	7.153	1.871	2.409	0.484	1.948
cpd21	1.483	0.749	0.760	0.631	15.963	9.216	8.123	1.965	4.062	-2.107	1.738
cpd22	1.391	0.610	0.699	0.673	17.747	9.185	6.996	2.537	1.652	-0.467	1.161
cpd23	1.712	0.559	0.664	0.659	14.482	11.846	8.659	1.673	6.038	-2.321	5.554
cpd24	1.66	0.549	0.617	0.633	20.356	12.282	7.371	2.762	8.754	2.767	-5.741
cpd25	1.757	0.431	0.593	0.586	20.663	11.005	8.829	2.341	5.62	-0.112	-3.292
cpd26	1.436	0.618	0.586	0.600	11.366	10.547	8.437	1.347	8.847	-2.657	7.587
cpd27	1.866	0.596	0.658	0.604	18.731	8.655	8.702	2.164	3.103	-0.237	0.589
cpd28	1.686	0.633	0.690	0.684	13.772	10.561	7.556	1.823	7.263	4.505	-3.457
cpd29	1.563	0.568	0.637	0.621	19.110	13.160	7.349	2.600	3.533	-2.802	1.541

cpd30	1.367	0.585	0.611	0.643	16.669	11.930	6.958	2.396	1.598	-0.869	0.485
cpd31	1.617	0.620	0.591	0.553	16.068	10.958	9.927	1.619	5.104	3.827	-3.357
cpd32	1.568	0.528	0.640	0.625	14.967	10.841	8.636	1.733	2.884	-2.318	-1.31
cpd33	1.836	0.550	0.614	0.653	15.057	11.141	8.693	1.732	3.173	1.782	2.605
cpd34	1.652	0.560	0.612	0.611	20.306	12.528	7.587	2.677	7.104	2.189	-5.754
cpd35	1.822	0.552	0.689	0.638	17.776	11.809	7.329	2.426	43	741.5	-36544
cpd36	1.511	0.642	0.644	0.615	11.946	9.845	7.343	1.627	26.8	20.4	-28715
Structures	Log[%DPPH]	X81	X82	X83	X84	X85	X86	X87			
cpd1	1.326	2.176	-7.783	-2.051	55.149	-5244.673	-42446	517.097			
cpd2	1.423	0.277	-7.806	-1.341	43.009	-3788.676	-29514	481.533			
cpd3	1.695	-4.603	-7.760	-0.974	47.676	-5345.689	-45436	28.808			
cpd4	1.483	-3.888	-7.696	-1.304	43.570	-4014.457	-30835	485.316			
cpd5	1.354	-5.065	-7.413	-1.085	39.915	-3762.835	-25295	373.922			
cpd6	1.389	-1.782	-7.554	-1.089	46.999	-4451.515	-36646	348.497			
cpd7	1.784	0.64	-7.642	-1.118	37.597	-3374.134	-24337	291.254			
cpd8	1.717	4.244	-7.641	-1.405	31.131	-3359.906	-21355	340.366			
cpd9	1.516	3.841	-8.038	-2.029	53.140	-6498.793	-51557	238.093			
cpd10	1.607	2.94	-8.103	-1.511	39.583	-5294.503	-36350	128.354			
cpd11	1.59	2.151	-7.882	-1.038	37.371	-5057.077	-38336	-126.794			
cpd12	1.412	-0.935	-8.218	-1.755	52.057	-6577.751	-53290	258.476			
cpd13	1.685	-8.844	-8.206	-1.601	41.780	-5268.373	-37960	211.016			
cpd14	1.571	0.289	-8.015	-1.544	38.692	-5017.131	-32750	90.852			
cpd15	1.428	-2.059	-7.445	-1.595	45.514	-5705.500	-44065	72.597			
cpd16	1.838	5.357	-8.219	-1.649	36.182	-4628.095	-32379	15.918			
cpd17	1.86	2.919	-7.810	-1.588	29.832	-4613.884	-27263	64.649			
cpd18	1.494	-4.643	-8.121	-1.617	38.595	-3628.355	-27236	269.030			
cpd19	1.646	-3.661	-7.915	-1.533	53.963	-5579.850	-43473	469.169			
cpd20	1.471	-1.33	-7.598	-1.462	44.009	-4044.157	-30978	430.234			
cpd21	1.483	-3.007	-7.607	-1.275	47.868	-4706.382	-40217	311.360			
cpd22	1.391	-1.078	-8.138	-1.2	37.119	-5284.645	-36009	-237.929			
cpd23	1.712	0.466	-7.769	-1.215	41.038	-5302.387	-36123	305.575			
cpd24	1.66	-6.001	-8.034	-1.954	55.885	-5499.938	-42128	470.780			
cpd25	1.757	4.553	-7.827	-1.353	43.918	-4270.187	-28948	428.282			
cpd26	1.436	-3.694	-8.059	-1.552	31.745	-3615.046	-23883	296.929			
cpd27	1.866	-3.037	-7.983	-1.666	40.758	-5549.669	-36992	84.315			
cpd28	1.686	4.528	-8.546	-1.941	37.489	-4882.571	-35326	-12.196			
cpd29	1.563	-1.504	-8.224	-1.656	53.022	-6833.782	-51758	194.504			
cpd30	1.367	-1.251	-7.733	-1.663	42.315	-5298.437	-36928	147.556			
cpd31	1.617	-0.373	-7.761	-1.24	46.823	-5960.788	-47895	25.754			
cpd32	1.568	-1.109	-7.382	-0.937	41.358	-4017.952	-29556	331.023			
cpd33	1.836	-0.328	-7.971	-1.559	39.992	-6555.914	-46349	40.236			
cpd34	1.652	-3.546	-8.040	-1.954	54.637	-6754.220	-50285	188.032			
cpd35	1.822	6.807	-8.121	-1.501	43.009	-5524.276	-36544	149.993			
cpd36	1.511	1.356	-8.203	-1.269	26.825	-4726.323	-28715	-274.813			

Table S4. Hyperparameters to be tested for (a) RF and (b) ANN.

(a) Random forest

Hyperparameter	**Value tested**
n estimators	30, 50, 70, 90
max depth	11, 12, 13, 14, 15
min samples split	1, 2, 3
min samples leaf	1, 2, 3

(b) Artificial neural networks

Hyperparameter	**Value tested**
number of hidden layers	1,2,3
number of neurons	2, 3, 4, 5, 6, 7, 8, 9
learning rate	0.01, 0.05, 0.1
batch size	4, 8, 16, 32
max_iter	600, 900, 1200, 1500

Table S5. Experimental and predicted Log[%DPPH] values for the training set and test set of NSAID-Se Derivatives obtained from RF, ANN models.

(a) Training set				(b) Training set			
No.	Experimental Log[%DPPH]	Predicted Log[%DPPH]		No.	Experimental Log[%DPPH]	Predicted Log[%DPPH]	
		RF_{RM}	ANN_{RM}			RF_{KS}	ANN_{KS}
1	1.326	1.434	1.390	1	1.326	1.344	1.329
2	1.423	1.462	1.430	3	1.695	1.640	1.748
3	1.695	1.616	1.682	6	1.389	1.406	1.390
4	1.483	1.518	1.472	7	1.784	1.706	1.779
5	1.354	1.437	1.341	9	1.516	1.551	1.521
6	1.389	1.419	1.368	10	1.607	1.698	1.648
9	1.516	1.561	1.570	11	1.590	1.541	1.590
10	1.607	1.661	1.537	12	1.412	1.433	1.372
13	1.685	1.638	1.643	13	1.685	1.654	1.718
14	1.571	1.526	1.562	15	1.428	1.468	1.412
15	1.428	1.456	1.321	16	1.838	1.749	1.823
16	1.838	1.729	1.827	17	1.860	1.794	1.748
17	1.860	1.724	1.821	18	1.494	1.503	1.490
18	1.494	1.527	1.441	19	1.646	1.595	1.748
19	1.646	1.595	1.564	20	1.471	1.467	1.463
20	1.471	1.450	1.424	21	1.483	1.492	1.481
21	1.483	1.498	1.522	22	1.391	1.428	1.383
22	1.391	1.471	1.330	23	1.712	1.646	1.701
24	1.660	1.628	1.689	24	1.660	1.625	1.650
25	1.757	1.726	1.768	26	1.436	1.473	1.428
28	1.686	1.730	1.672	28	1.686	1.717	1.699
29	1.563	1.661	1.670	29	1.563	1.579	1.552
30	1.367	1.500	1.427	30	1.367	1.388	1.383
31	1.617	1.656	1.721	31	1.617	1.641	1.601
32	1.568	1.511	1.469	32	1.568	1.487	1.582
33	1.836	1.747	1.778	33	1.836	1.725	1.748
34	1.652	1.620	1.691	34	1.652	1.611	1.651
35	1.822	1.740	1.812	36	1.511	1.611	1.520
Test set				**Test set**			
7	1.784	1.693	1.828	2	1.423	1.478	1.508
8	1.717	1.711	1.637	4	1.483	1.511	1.424
11	1.590	1.564	1.515	5	1.354	1.450	1.377
12	1.412	1.522	1.361	8	1.717	1.665	1.748
23	1.712	1.661	1.657	14	1.571	1.590	1.473
26	1.436	1.576	1.557	25	1.757	1.705	1.748
27	1.866	1.742	1.783	27	1.866	1.719	1.748
36	1.511	1.589	1.576	35	1.822	1.758	1.868

Figure S1. Comparison of predicted Log[%DPPH] for new NSAIDs-Se derivatives based on RF_{KS} and ANN_{KS} models, and experimental results.

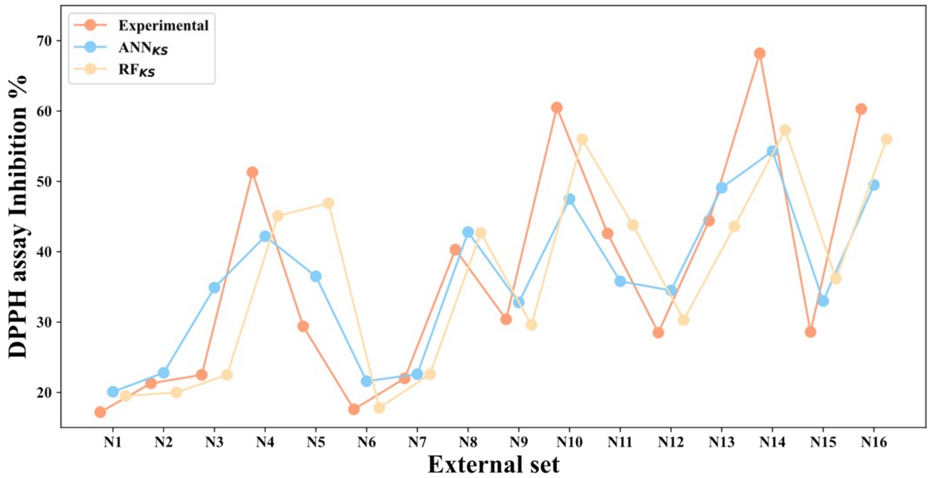

Table S6. Five molecular descriptors of 16 new NSAIDs-Se derivatives.

No.	E-state keys (sums): S_dssC	E-state keys (sums): S_sCH3	Dipole y	HOMO eigenvalue	Heat of formation
N1	0.051	6.379	1.066	-7.554	333.633
N2	-0.122	1.849	-1.415	-7.582	518.979
N3	0.015	3.591	6.948	-7.562	411.175
N4	-0.330	0.000	-3.88	-7.627	395.215
N5	-0.134	1.785	4.661	-7.670	448.962
N6	3.451	4.537	3.214	-7.532	405.688
N7	-0.105	4.157	4.715	-7.522	484.389
N8	0.407	1.922	1.594	-7.645	370.181
N9	0.649	7.964	1.122	-7.681	305.544
N10	0.440	3.422	1.589	-7.628	486.253
N11	0.604	5.174	-3.538	-7.594	379.411
N12	0.191	1.544	4.725	-7.630	366.458
N13	0.428	3.351	1.624	-7.768	410.715
N14	2.958	5.865	-0.862	-7.451	372.215
N15	0.479	5.741	0.724	-7.642	474.437
N16	0.987	3.506	1.427	-7.758	338.760

Code S1. Python codes optimization of the QSAR models

Generate RF Model from Dataset Split by KS

```
# Read the CSV file
da = pd.read_csv('/Users/pipi/54-5des.csv')

# Split the data into training (train), testing (test), and external (ext) sets
train = da.head(28)
test = da.iloc[28:36]
ext = da.tail(16)

# Define the target columns
y0_kstrain = train.iloc[:, 0]
y0_kstest = test.iloc[:, 0]
y0_ksext = ext.iloc[:, 0]

# Define the feature columns for Random Forest
X_train_rf = train.iloc[:, 2:]
X_test_rf = test.iloc[:, 2:]
X_ext_rf = ext.iloc[:, 2:]

# Define the hyperparameters to be tested for Random Forest
param_grid_rf = {
    'n_estimators': [30, 50, 70, 90],
    'max_depth': [11, 12, 13, 14, 15],
    'min_samples_split': [1, 2, 3],
    'min_samples_leaf': [1, 2, 3]}
rf_model = RandomForestRegressor()

# Use GridSearchCV to search for the best hyperparameter combination
grid_search_rf = GridSearchCV(estimator=rf_model, param_grid=param_grid_rf,
                              scoring='neg_mean_squared_error', cv=5)
grid_search_rf.fit(X_train_rf, y0_kstrain)

# Get the best hyperparameters from the grid search
best_params_rf = grid_search_rf.best_params_

# Train a Random Forest model with the best hyperparameters
best_rf_model = RandomForestRegressor(**best_params_rf)

# Internal validation using leave-one-out cross-validation
from sklearn.model_selection import LeaveOneOut
```

```
loo = LeaveOneOut()

# Initialize a list to store predictions
y_pred_rf_loo = []

# Perform leave-one-out cross-validation
for train_index, test_index in loo.split(X_train_rf):
    X_train_loo, X_test_loo = X_train_rf.iloc[train_index], X_train_rf.iloc[test_index]
    y_train_loo, y_test_loo = y0_kstrain.iloc[train_index], y0_kstrain.iloc[test_index]

    # Train a Random Forest model
    rf_model_loo = RandomForestRegressor(**best_params_rf)
    rf_model_loo.fit(X_train_loo, y_train_loo)

    # Make predictions on the left-out sample
    y_pred_loo = rf_model_loo.predict(X_test_loo)

    # Store the prediction
    y_pred_rf_loo.extend(y_pred_loo)

# Store leave-one-out predictions in the DataFrame
KS_rf['Predicted Values (LOO)'] = y_pred_rf_loo

# Evaluate the model performance on leave-one-out cross-validation
r2_rf_loo = r2_score(y0_rmtrain, y_pred_rf_loo)
rmse_rf_loo = np.sqrt(mean_squared_error(y0_rmtrain, y_pred_rf_loo))

# Output results for leave-one-out cross-validation
print("\nRandom Forest Performance on Leave-One-Out Cross-Validation:")
print("R^2:", r2_rf_loo)
print("RMSE:", rmse_rf_loo)

best_rf_model.fit(X_train_rf, y0_kstrain)

# Make predictions on the test set
y_pred_rf_test = best_rf_model.predict(X_test_rf)

# Evaluate the model performance on the test set
r2_rf_test = r2_score(y0_kstest, y_pred_rf_test)
rmse_rf_test = np.sqrt(mean_squared_error(y0_kstest, y_pred_rf_test))
```

```python
# Make predictions on the external set
y_pred_rf_ext = best_rf_model.predict(X_ext_rf)

# Evaluate the model performance on the external set
r2_rf_ext = r2_score(y0_ksext, y_pred_rf_ext)
rmse_rf_ext = np.sqrt(mean_squared_error(y0_ksext, y_pred_rf_ext))


# Output results
print("Best Hyperparameters (Random Forest):", best_params_rf)
print("\nRandom Forest Performance on Test Set:")
print("R^2:", r2_rf_test)
print("RMSE:", rmse_rf_test)

print("\nRandom Forest Performance on External Set:")
print("R^2:", r2_rf_ext)
print("RMSE:", rmse_rf_ext)

# Output the DataFrame with predictions and true values
print("KS_RF DataFrame:")
print(KS_rf)
```

Generate RF Model from Dataset Split by RM

```
da = pd.read_csv('/Users/pipi/54-5des.csv')
# Split the data into training (train), testing (test), and external (ext) sets
train = da.head(28)
test = da.iloc[28:36]
ext = da.tail(16)
# Define the target columns
y0_rmtrain = train.iloc[:, 1]
y0_rmtest = test.iloc[:, 1]
y0_rmext = ext.iloc[:, 1]
# Define the feature columns for Random Forest
X_train_rf = train.iloc[:, 2:]
X_test_rf = test.iloc[:, 2:]
X_ext_rf = ext.iloc[:, 2:]
# Define the hyperparameters to be tested for Random Forest
param_grid_rf = {
    'n_estimators': [30, 50, 70, 90],
    'max_depth': [11, 12, 13, 14, 15],
    'min_samples_split': [1, 2, 3],
    'min_samples_leaf': [1, 2, 3]}
# Initialize the Random Forest model
rf_model = RandomForestRegressor()
# Use GridSearchCV to search for the best hyperparameter combination
grid_search_rf = GridSearchCV(estimator=rf_model, param_grid=param_grid_rf,
                              scoring='neg_mean_squared_error', cv=5)
grid_search_rf.fit(X_train_rf, y0_rmtrain)

# Get the best hyperparameters from the grid search
best_params_rf = grid_search_rf.best_params_

# Train a Random Forest model with the best hyperparameters
best_rf_model = RandomForestRegressor(**best_params_rf)

# Internal validation using leave-one-out cross-validation
from sklearn.model_selection import LeaveOneOut

loo = LeaveOneOut()

# Initialize a list to store predictions
y_pred_rf_loo = []
# Perform leave-one-out cross-validation
for train_index, test_index in loo.split(X_train_rf):
    X_train_loo, X_test_loo = X_train_rf.iloc[train_index], X_train_rf.iloc[test_index]
    y_train_loo, y_test_loo = y0_rmtrain.iloc[train_index], y0_rmtrain.iloc[test_index]

    # Train a Random Forest model
    rf_model_loo = RandomForestRegressor(**best_params_rf)
    rf_model_loo.fit(X_train_loo, y_train_loo)


    # Make predictions on the left-out sample
    y_pred_loo = rf_model_loo.predict(X_test_loo)

    # Store the prediction
    y_pred_rf_loo.extend(y_pred_loo)

# Store leave-one-out predictions in the DataFrame
rm_rf['Predicted Values (LOO)'] = y_pred_rf_loo

# Evaluate the model performance on leave-one-out cross-validation
r2_rf_loo = r2_score(y0_rmtrain, y_pred_rf_loo)
rmse_rf_loo = np.sqrt(mean_squared_error(y0_rmtrain, y_pred_rf_loo))

# Output results for leave-one-out cross-validation
print("\nRandom Forest Performance on Leave-One-Out Cross-Validation:")
print("R^2:", r2_rf_loo)
print("RMSE:", rmse_rf_loo)
```

```
best_rf_model.fit(X_train_rf, y0_rmtrain)

# Make predictions on the test set
y_pred_rf_test = best_rf_model.predict(X_test_rf)

# Evaluate the model performance on the test set
r2_rf_test = r2_score(y0_rmtest, y_pred_rf_test)
rmse_rf_test = np.sqrt(mean_squared_error(y0_rmtest, y_pred_rf_test))

# Make predictions on the external set
y_pred_rf_ext = best_rf_model.predict(X_ext_rf)

# Evaluate the model performance on the external set
r2_rf_ext = r2_score(y0_rmext, y_pred_rf_ext)
rmse_rf_ext = np.sqrt(mean_squared_error(y0_rmext, y_pred_rf_ext))

# Output results
print("Best Hyperparameters (Random Forest):", best_params_rf)
print("\nRandom Forest Performance on Test Set:")
print("R^2:", r2_rf_test)
print("RMSE:", rmse_rf_test)

print("\nRandom Forest Performance on External Set:")
print("R^2:", r2_rf_ext)
print("RMSE:", rmse_rf_ext)

# Output the DataFrame with predictions and true values
print("rm_RF DataFrame:")
print(rm_rf)
```

Generate ANN Model from Dataset Split by KS

```
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import GridSearchCV, cross_val_predict, LeaveOneOut
from sklearn.metrics import mean_squared_error, r2_score
da = pd.read_csv('/Users/pipi/54-5des.csv')
# Split the data into training (train), testing (test), and external (ext) sets
train = da.head(28)
test = da.iloc[28:36]
ext = da.tail(16)
# Define the target columns
y0_kstrain = train.iloc[:, 0]
y0_kstest = test.iloc[:, 0]
y0_ksext = ext.iloc[:, 0]
# Define the feature columns for Artificial Neural Network (ANN)
X_train_ann = train.iloc[:, 2:]
X_test_ann = test.iloc[:, 2:]
X_ext_ann = ext.iloc[:, 2:]

# Define the DataFrame to store predictions and true values
KS_ANN = pd.DataFrame()

# Define the hyperparameters to be tested for ANN
param_grid_ann = {
    'hidden_layer_sizes': [(i,) for i in range(2, 9)],  # 1 to 3 hidden layers
    'learning_rate_init': [0.01, 0.05, 0.1],  # initial learning rate
    'batch_size': [4, 8, 16, 32],
    'max_iter': [600, 900, 1200, 1500]}
ann_model = MLPRegressor()

# Use GridSearchCV to search for the best hyperparameter combination with 5-fold cross-validation
grid_search_ann = GridSearchCV(estimator=ann_model, param_grid=param_grid_ann,
                               scoring='neg_mean_squared_error', cv=5)
grid_search_ann.fit(X_train_ann, y0_ks
                    train)

# Get the best hyperparameters from the grid search
best_params_ann = grid_search_ann.best_params_

# Train an ANN model with the best hyperparameters
best_ann_model = MLPRegressor(**best_params_ann)

# Internal validation using leave-one-out cross-validation
loo = LeaveOneOut()

# Initialize a list to store predictions
y_pred_ann_loo = []
```

```
# Perform leave-one-out cross-validation
for train_index, test_index in loo.split(X_train_ann):
    X_train_loo, X_test_loo = X_train_ann.iloc[train_index], X_train_ann.iloc[test_index]
    y_train_loo, y_test_loo = y0_kstrain.iloc[train_index], y0_kstrain.iloc[test_index]

    # Train an ANN model
    ann_model_loo = MLPRegressor(**best_params_ann)
    ann_model_loo.fit(X_train_loo, y_train_loo)

    # Make predictions on the left-out sample
    y_pred_loo = ann_model_loo.predict(X_test_loo)

    # Store the prediction
    y_pred_ann_loo.extend(y_pred_loo)

# Store leave-one-out predictions in the DataFrame
KS_ANN['Predicted Values (LOO)'] = y_pred_ann_loo

# Evaluate the model performance on leave-one-out cross-validation
r2_ann_loo = r2_score(y0_kstrain, y_pred_ann_loo)
rmse_ann_loo = np.sqrt(mean_squared_error(y0_kstrain, y_pred_ann_loo))

# Make predictions on the test set
y_pred_ann_test = ann_model_loo.predict(X_test_ann)

# Store predictions and true values in the DataFrame
KS_ANN['True Values'] = y0_kstest
KS_ANN['Predicted Values (ANN)'] = y_pred_ann_test
```

```
# Evaluate the model performance on the test set
r2_ann_test = r2_score(y0_kstest, y_pred_ann_test)
rmse_ann_test = np.sqrt(mean_squared_error(y0_kstest, y_pred_ann_test))

# Make predictions on the external set
y_pred_ann_ext = ann_model_loo.predict(X_ext_ann)

# Store external set predictions in the DataFrame
KS_ANN['Predicted Values (ANN External)'] = y_pred_ann_ext

# Evaluate the model performance on the external set
r2_ann_ext = r2_score(y0_ksext, y_pred_ann_ext)
rmse_ann_ext = np.sqrt(mean_squared_error(y0_ksext, y_pred_ann_ext))


# Output the DataFrame with predictions and true values
print("KS_ANN DataFrame:")
print(KS_ANN)

# Output results
print("\nBest Hyperparameters (ANN):", best_params_ann)
print("\nANN Performance on Leave-One-Out Cross-Validation:")
print("R^2:", r2_ann_loo)
print("RMSE:", rmse_ann_loo)

print("\nANN Performance on Test Set:")
print("R^2:", r2_ann_test)
print("RMSE:", rmse_ann_test)

print("\nANN Performance on External Set:")
print("R^2:", r2_ann_ext)
print("RMSE:", rmse_ann_ext)
```

Generate ANN Model from Dataset Split by RM

```
import pandas as pd
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import GridSearchCV, cross_val_predict, LeaveOneOut
from sklearn.metrics import mean_squared_error, r2_score
# Read the CSV file
da = pd.read_csv('/Users/pipi/54-5des.csv')
# Split the data into training (train), testing (test), and external (ext) sets
train = da.head(28)
test = da.iloc[28:36]
ext = da.tail(16)
# Define the target columns
y0_rmtrain = train.iloc[:, 1]
y0_rmtest = test.iloc[:, 1]
y0_rmext = ext.iloc[:, 1]
# Define the feature columns for Artificial Neural Network (ANN)
X_train_ann = train.iloc[:, 2:]
X_test_ann = test.iloc[:, 2:]
X_ext_ann = ext.iloc[:, 2:]
# Define the DataFrame to store predictions and true values
KS_ANN = pd.DataFrame()

# Define the hyperparameters to be tested for ANN
param_grid_ann = {
    'hidden_layer_sizes': [(i,) for i in range(2, 9)],  # 1 to 3 hidden layers
    'learning_rate_init': [0.01, 0.05, 0.1],  # initial learning rate
    'batch_size': [4, 8, 16, 32],
    'max_iter': [600, 900, 1200, 1500]
}

# Initialize the ANN model
ann_model = MLPRegressor()

# Use GridSearchCV to search for the best hyperparameter combination with 5-fold cross-validation
grid_search_ann = GridSearchCV(estimator=ann_model, param_grid=param_grid_ann,
                               scoring='neg_mean_squared_error', cv=5)
grid_search_ann.fit(X_train_ann, y0_kstrain)
# Get the best hyperparameters from the grid search
best_params_ann = grid_search_ann.best_params_
# Train an ANN model with the best hyperparameters
best_ann_model = MLPRegressor(**best_params_ann)
# Internal validation using leave-one-out cross-validation
loo = LeaveOneOut()
# Initialize a list to store predictions
y_pred_ann_loo = []

# Perform leave-one-out cross-validation
for train_index, test_index in loo.split(X_train_ann):
    X_train_loo, X_test_loo = X_train_ann.iloc[train_index], X_train_ann.iloc[test_index]
    y_train_loo, y_test_loo = y0_rmtrain.iloc[train_index], y0_rmtrain.iloc[test_index]

    # Train an ANN model
    ann_model_loo = MLPRegressor(**best_params_ann)
    ann_model_loo.fit(X_train_loo, y_train_loo)

    # Make predictions on the left-out sample
    y_pred_loo = ann_model_loo.predict(X_test_loo)

    # Store the prediction
    y_pred_ann_loo.extend(y_pred_loo)

# Store leave-one-out predictions in the DataFrame
RM_ANN['Predicted Values (LOO)'] = y_pred_ann_loo

# Evaluate the model performance on leave-one-out cross-validation
r2_ann_loo = r2_score(y0_kstrain, y_pred_ann_loo)
rmse_ann_loo = np.sqrt(mean_squared_error(y0_kstrain, y_pred_ann_loo))

# Make predictions on the test set
y_pred_ann_test = ann_model_loo.predict(X_test_ann)

# Store predictions and true values in the DataFrame
KS_ANN['True Values'] = y0_kstest
KS_ANN['Predicted Values (ANN)'] = y_pred_ann_test

# Evaluate the model performance on the test set
r2_ann_test = r2_score(y0_kstest, y_pred_ann_test)
rmse_ann_test = np.sqrt(mean_squared_error(y0_kstest, y_pred_ann_test))
```

```
# Make predictions on the external set
y_pred_ann_ext = ann_model_loo.predict(X_ext_ann)

# Store external set predictions in the DataFrame
RM_ANN['Predicted Values (ANN External)'] = y_pred_ann_ext

# Evaluate the model performance on the external set
r2_ann_ext = r2_score(y0_rmext, y_pred_ann_ext)
rmse_ann_ext = np.sqrt(mean_squared_error(y0_rmext, y_pred_ann_ext))

# Output the DataFrame with predictions and true values
print("RM_ANN DataFrame:")
print(RM_ANN)

# Output results
print("\nBest Hyperparameters (ANN):", best_params_ann)
print("\nANN Performance on Leave-One-Out Cross-Validation:")
print("R^2:", r2_ann_loo)
print("RMSE:", rmse_ann_loo)

print("\nANN Performance on Test Set:")
print("R^2:", r2_ann_test)
print("RMSE:", rmse_ann_test)

print("\nANN Performance on External Set:")
print("R^2:", r2_ann_ext)
print("RMSE:", rmse_ann_ext)
```

Y-scrambling Test

```
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.utils import shuffle
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Initialize lists to store results
r2_results = []
r2_cv_results = []
rmse_results = []

# Perform the Y-scrambling experiment 10 times
for _ in range(10):
    # Shuffle the target values to scramble the relationship with predictors
    y_scrambled = shuffle(train_Y, random_state=42)

    # Fit the best model to the scrambled data
    best_model.fit(train_X, y_scrambled)

    # Predict using the fitted model
    y_scrambled_pred = best_model.predict(train_X)

    # Calculate RMSE and R^2 for the scrambled predictions
    rmse = np.sqrt(mean_squared_error(y_scrambled, y_scrambled_pred))
    r2 = r2_score(y_scrambled, y_scrambled_pred)

    # Perform cross-validation and calculate the average R^2 score
    r2_cv_scores = cross_val_score(best_model, train_X, y_scrambled, cv=3, scoring='r2')
    r2_cv_avg = np.mean(r2_cv_scores)

    # Append the results to the respective lists
    rmse_results.append(rmse)
    r2_results.append(r2)
    r2_cv_results.append(r2_cv_avg)

# Print the results of the Y-scrambling experiment
print("Y-scrambling Experiment Results:")
print("R^2 Results: ", [f'{r2:.4f}' for r2 in r2_results])
print("R²cv Results: ", [f'{r2_cv:.4f}' for r2_cv in r2_cv_results])
print("Average R^2: ", f'{np.mean(r2_results):.4f}')
print("Average R²cv: ", f'{np.mean(r2_cv_results):.4f}')
```